目录

[1、PHP服务缓存加速优化实战 1](#_Toc441060668)

[1.1 PHP缓存加速器介绍与环境准备 1](#_Toc441060669)

[1.1.1 PHP缓存加速器介绍 1](#_Toc441060670)

[1.1.2 LAMP环境PHP缓存加速器原理 2](#_Toc441060671)

[1.1.3 LNMP环境PHP缓存加速器原理详解 2](#_Toc441060672)

[1.1.4 缓存加速器软件种类及选择建议 2](#_Toc441060673)

[1.1.5 PHP缓存加速器安装环境准备 3](#_Toc441060674)

[1.1.6 有关LNMP环境扩展软件的部署说明 4](#_Toc441060675)

[1.2 安装PHP缓存加速器扩展 5](#_Toc441060676)

[1.2.1 安装PHP eaccelerator缓存加速模块 5](#_Toc441060677)

[1.2.2 安装PHP xcache缓存加速模块 6](#_Toc441060678)

[1.2.3 APC缓存及加速插件说明 7](#_Toc441060679)

[1.2.4 PHP官方插件zendopcache 7](#_Toc441060680)

[1.3 安装数据库缓存及其他PHP扩展插件 7](#_Toc441060681)

[1.3.1 安装PHP memcached扩展插件 7](#_Toc441060682)

[1.3.2 安装PDO\_MYSQL扩展模块 9](#_Toc441060683)

[1.4 安装其他的PHP扩展插件模块 10](#_Toc441060684)

[1.4.1 安装图像处理程序及imagick扩展模块 10](#_Toc441060685)

[1.4.2 检查所有PHP扩展插件模块安装的成果 12](#_Toc441060686)

[1.5 配置PHP加速与缓存相关扩展插件模块 12](#_Toc441060687)

[1.5.1 配置memcache/pdo\_mysql/imagick模块生效 12](#_Toc441060688)

[1.5.2 配置eaccelerator插件生效并优化参数 14](#_Toc441060689)

[1.5.3 配置xcache插件加速 17](#_Toc441060690)

[1.5.4 配置zendopcache插件加速 19](#_Toc441060691)

[1.6 生产环境PHP扩展插件的安装建议 21](#_Toc441060692)

[1.6.1 PHP的安装插件表格列表 21](#_Toc441060693)

[1.6.2 生产环境插件的安装建议 21](#_Toc441060694)

[1.6.3 PHP加速插件的测试对比 21](#_Toc441060695)

[1.7 补充一些知识 22](#_Toc441060696)

[1.7.1 phpize是什么？ 22](#_Toc441060697)

[1.7.2 php指定MySQL的编译参数带来的问题 22](#_Toc441060698)

# 1、PHP服务缓存加速优化实战

## 1.1 PHP缓存加速器介绍与环境准备

### 1.1.1 PHP缓存加速器介绍

##### 1、操作码介绍及缓存原理

当客户端请求一个PHP程序时，服务器的PHP引擎会解析该PHP程序，并将其编译为特定的操作码文件（operate code，opcode），这是要执行的PHP代码的一种二进制表示形式。默认情况下，这个编译好的操作码文件由PHP引擎执行后丢弃。操作码缓存（opcode cache）的原理就是将这个编译后的操作码保存下来，并放入到共享内存里，以便在下一次调用该PHP页面时重用它，避免了相同代码的重复编译。节省了PHP引擎重复编译的时间，降低了服务器负载，同时减少了CPU和内存开销。

##### 2. PHP缓存加速软件介绍

为了提高PHP引擎的高并发访问及执行速度，产生了一系列PHP缓存及加速软件。这些软件设计的目的就是缓存前文提到的PHP引擎解析过的操作码文件，此时，在指定时间内若有相同的PHP程序请求访问，就不需要重复解析编译了，而是直接调用缓存中的PHP操作码文件，这样就提高了动态web服务的处理速度，从而提升了用户访问企业网站的整体体验。

### 1.1.2 LAMP环境PHP缓存加速器原理

在lamp环境中，Apache服务使用的是libphp5.so相应处理PHP程序请求的，整个流程大概如下：

1） Apache接收客户的PHP程序请求并根据规则过滤之。

2） Apache将PHP程序请求传递给PHP处理模块libphp5.so。

3） PHP引擎定位磁盘上的PHP文件，并将其加载到内存中解析。

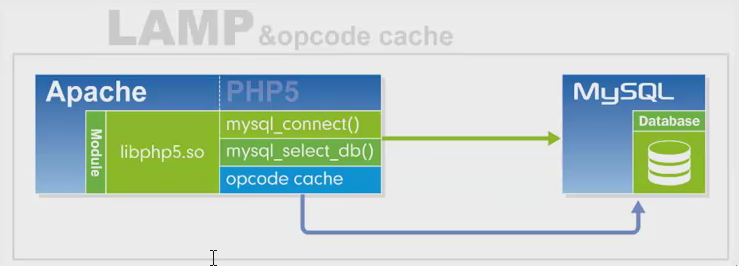
4） PHP处理模块libphp5.so强PHP源代码编译成opcode。

5） PHP处理模块libphp5.so执行opcode，然后把opcode缓存起来。

6） Apache接收客户端新的PHP程序请求，PHP引擎直接读取缓存执行opcode文件，并将结果返回。此时，就无第4步的编译解析了，从而提升了PHP编译解析效率。

PHP缓存加速器解决的是上述第5步的问题，默认情况下PHP会将opcode内容执行后丢弃，这里却通过PHP缓存加速软件，将opcode内容缓存了下来，目的是当PHP重复请求时，不需要再重复编译解析PHP程序代码，因为在高并发高访问量的网站上，大量的重复编译会消耗很多系统资源和时间，这里的瓶颈就是PHP的重复编译，即影响了处理速度，又加重了服务器的负载负担，为了解决此问题，PHP缓存加速器就这样诞生了。

下图是lamp环境下PHP请求及操作码缓存过程的原理示意图。



上图经典网站架构PHP操作码缓存原理示意图

### 1.1.3 LNMP环境PHP缓存加速器原理详解

在lnmp环境中，PHP引擎不再使用libphp5.so模块了，而是启动了独立的FCGI（PHP-fpm）进程，由它监听来自nginx的PHP程序请求，并交给PHP引擎解析处理，整个执行流程大概如下：

1） nginx接收客户端的PHP程序访问请求。

2） nginx根据扩展名等过滤规则将PHP程序请求传递解析PHP的fcgi进程（PHP-FPM）。

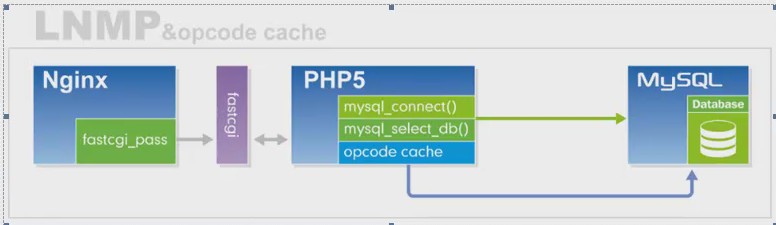
3） PHP-FPM进程调用PHP解析器读取站点磁盘上的PHP文件，并加载到内存中。

4） PHP解析器将PHP程序编译称为opcode文件，然后把opcode缓存起来。

5） PHP FPM引擎执行opcode树，返回数据给nginx，进而返回客户端。

6） nginx接收客户新的PHP程序请求，PHP FPM引擎就会直接读取缓存中的opcode执行，将结果返回，而无需第4步操作，从而提升了PHP编译解析效率。

下图为LNMP环境下PHP请求操作码缓存过程的原理示意图。



### 1.1.4 缓存加速器软件种类及选择建议

PHP缓存加速器软件常见的种类有xcache、eaccelerator、APC（alternative PHP cache），zendopcache等，在企业环境我们要如何选择PHP缓存加速器软件呢？

事实上，任选其一即可，没必要都安装上，都安装也可能会发生冲突。总的选择建议就是根据企业的业务需求及选择前的压力测试结果，或者根据个人的经验偏好选择。不过，建立首选xcache，其次是eaccelerator，如果想尝新，可以选择zendopcache，限于篇幅原因此次未对APC加速软件做深入讲解。

###### 首选xcache的原因如下：

* 经过测试xcache效率更好，更快。
* xcache软件开发社区更活跃，最新版2014年底发布。
* 支持更高版本的PHP，例如PHP5.5、PHP5.6。

###### 次选eaccelerator的原因如下：

* 安装及配置参数更简单、减速效果也不错。
* 文档资料较多，但官方对软件的更新很慢，社区不活跃。
* 仅适合PHP版本5.4以下的程序。

###### 选择zendopcache的原因如下：

* 是PHP官方研发的新一代缓存加速软件，以后的发展潜力可能会很好，PHP5.5以前的版本可以通过zendopcache软件以插件扩展的方式安装，从PHP5.5版本开始已经整合到PHP软件里了，编译时只需指定一个参数即可，例如：

--enable-opcache。

* zendopcache可能是未来的缓存加速首选，现在的稳定性还有待检验，小规模环境下PHP5以前的版本可以通过插件式安装使用，PHP5以上可以直接指定参数编译使用，若可以忍受zendopcache的各种未知问题的话，也可以尝试使用。

### 1.1.5 PHP缓存加速器安装环境准备

##### 1.LNMP 基础web环境准备

在安装PHP的扩展及缓存加速软件之前，需要先安装好LNMP的完整环境，例如：能配置出现PHPinfo信息的界面表示PHP服务正常安装，同时最好可以编写一个调用数据库的简单PHP程序，例如：test\_mysql.php，进而确认MySQL数据库是不是正常的。

当LNMP环境软件的各个版本信息如下表所示。

下表LNMP环境软件及版本说明：

|  |  |
| --- | --- |
| **软件** | **版本** |
| Linux | Centos6.6 64bit |
| Nginx | 1.6.3 |
| MySQL | 5.5.32 64bit |
| PHP | 5.3.27 |

如果上述软件的版本对不上，在安装PHP的扩展软件时有可能会遇到一些小问题。因此，建议在使用过程中和本文档给出的版本保持一致，操作也要保持一致，否则可能会出现额外的问题，影响学习进度，等按照文档上的操作完成了部署后，再去变换版本操作。这样的学习方法是最好的，也是最佳的方式。

##### 2. 检查LNMP的软件版本

（1） 查看Linux内核及版本相关信息：

[root@lnmp ~]# cat /etc/redhat-release

CentOS release 6.7 (Final)

[root@lnmp ~]# uname -r

2.6.32-573.el6.x86\_64

[root@lnmp ~]# uname -m

x86\_64

（2） 查看nginx web版本相关信息：

[root@lnmp ~]# /application/nginx/sbin/nginx -v

nginx version: nginx/1.6.3

（3） 查看PHP服务版本相关信息：

[root@lnmp ~]# /application/php/bin/php -v

PHP 5.3.27 (cli) (built: Dec 4 2015 13:23:55)

Copyright (c) 1997-2013 The PHP Group

Zend Engine v2.3.0, Copyright (c) 1998-2013 Zend Technologies

（4） 查看MySQL服务版本相关信息：

[root@db01 ~]# mysqladmin -uroot -poldboy123 version

mysqladmin Ver 8.42 Distrib 5.5.32, for linux2.6 on x86\_64

Copyright (c) 2000, 2013, Oracle and/or its affiliates. All rights reserved.

Oracle is a registered trademark of Oracle Corporation and/or its

affiliates. Other names may be trademarks of their respective

owners.

Server version 5.5.32-log

Protocol version 10

Connection Localhost via UNIX socket

UNIX socket /tmp/mysql.sock

Uptime: 49 sec

Threads: 1 Questions: 1 Slow queries: 0 Opens: 35 Flush tables: 1 Open tables: 4 Queries per second avg: 0.020

### 1.1.6 有关LNMP环境扩展软件的部署说明

##### 1. LNMP缓存加速特别提示

不管是Apache还是nginx，最后都是通过PHP提供动态查询解析的，因此，不管是Apache的libphp5.so模块方式，还是nginx的FCGI的PHP服务方式，最终在PHP引擎上的优化也一致的，即都是基于PHP服务（php.ini）的，因此，如无特殊说明，本节以后的环境安装和优化均使用与LNMP和LAMP。

##### 2. 解决部分加速软件的Perl编译问题。

在下面各类软件的安装编译过程中，如果不解决Perl的一些环境问题可能会带来意想不到的安装错误或警告，为了避免出现这些问题导致前功尽弃，下面会把一些影响问题的拦路虎提前说明，请解决之。

（1） 配置环境变量LC\_ALL

配置环境变量LC\_ALL的过程如下：

[root@db01 ~]# echo 'export LC\_ALL=C' >> /etc/profile

#设置环境变量，解决后面Perl出现插件的编译问题。符号“>>”表示向文件追加内容。

[root@db01 ~]# tail -1 /etc/profile

export LC\_ALL=C

#查看是否正确追加了export LC\_ALL=C 环境配置。

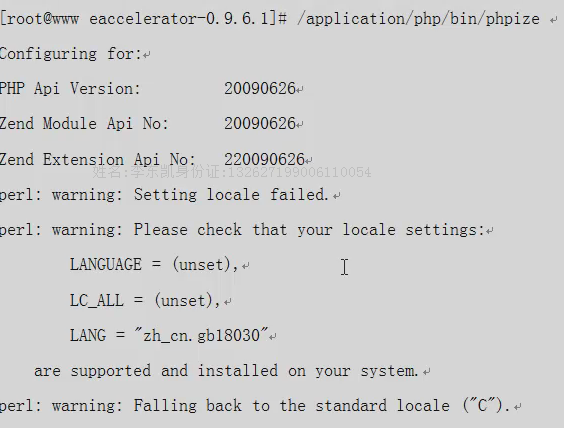
[root@db01 ~]# source /etc/profile

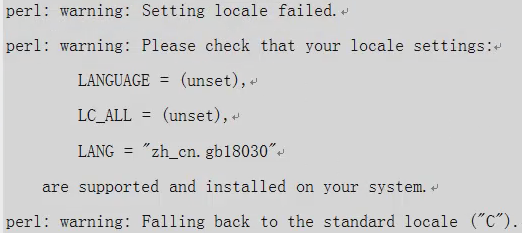
#使增加的环境变量配置生效。

[root@db01 ~]# echo $LC\_ALL

C

如果不设置该变量，在安装某加速软件时，可能会遇到如下的警告问题（安装eaccelerator时遇到的警告）：





（2） 安装Perl相关软件依赖。

需要提前安装Perl相关软件依赖软件包，执行yum install --y perl-CPAN或yum install --y perl-devel，任意一个即可，大约依赖17个包。

如果不安装上述软件包，在后面安装ImageMagick时可能会报错，后文安装ImageMagick时有相应的报错说明。

## 1.2 安装PHP缓存加速器扩展

### 1.2.1 安装PHP eaccelerator缓存加速模块

##### 1. eaccelerator缓存加速插件说明

eaccelerator是一个免费的、开放源代码的PHP加速、优化以及缓存的扩展插件软件，它可以缓存PHP程序编译后的中间代码文件（opcode）、session数据等，减低PHP程序在编译解析时对服务器的性能开销。eaccelerator还可以加快PHP程序的执行速度，降低服务器负载压力，使PHP程序代码执效率能提高1~10倍。

eaccelerator会把编译好的PHP程序存放在共享内存里，然后每次从内存里调用执行，可以设定把一些不适合放内存里缓存的编译结果存储到磁盘上，默认情况，磁盘和内存缓存都会被eaccelerator使用。

eaccelerator诞生于2004年，前身是turck MMCache，因此开发者进入了zend公司工作，所以开发eaccelerator的人继承了Turck MMCache的一些特性，从而设计出了eaccelerator加速器。

eaccelerator算是一个老牌的缓存加速软件，曾经在结合PHP引擎解析时被广泛使用，成熟稳定，目前代码更新不活跃，因此，使用的企业逐渐减少，但eaccelerator仍是一款值得信赖的缓存加速软件。XCache的官方也称赞eaccelerator是不错的opcode缓存器。

eaccelerator的最新版为0.9.6.1，支持的PHP最新版本为PHP5.3以及以前5系列的版本。

早期的0.9.5版本支持PHP4和PHP5.2及以前的版本。

eaccelerator最新版0.9.6.1版的下载地址为：

<https://github.com/eaccelerator/eaccelerator/downloads>

##### 2. eaccelerator插件安装过程

具体的安装目录集如下：

[root@lnmp tools]# tar xf eaccelerator-0.9.6.1.tar.bz2

[root@lnmp tools]# cd eaccelerator-0.9.6.1

[root@lnmp eaccelerator-0.9.6.1]# /application/php/bin/phpize

[root@lnmp eaccelerator-0.9.6.1]# ./configure --enable-eaccelerator=shared --with-php-config=/application/php/bin/php-config

###===configure的参数路径要正确配置，特别是后面的--with-php-config参数对应的PHP安装目录地址，几乎所有的PHP扩展都要指定这个参数。

[root@lnmp eaccelerator-0.9.6.1]#make

[root@lnmp eaccelerator-0.9.6.1]#make install

[root@lnmp eaccelerator-0.9.6.1]#cd ../

[root@lnmp tools]# ls /application/php5.3.27/lib/php/extensions/no-debug-non-zts-20090626/

eaccelerator.so

下面是实践得出的注意事项及说明：

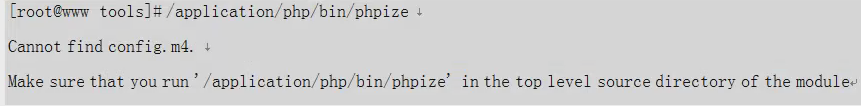
1） PHP5.3.XX可用eaccelerator-0.9.6版本，如果使用0.9.5.2版本在make阶段会报如下错误：make：\*\*\*[optimize.lo]Error 1.

2） PHP5.2.XX可用eaccelerator-0.9.5.2版本。

提示：上述测试验证了eaccelerator的官方说明，请选择时注意。

下面试之前遇到的问题故障。

问题1：cannot find config.m4.错误



解答：需要切到eaccelerator路径下执行/application/php/bin/phpize.

### 1.2.2 安装PHP xcache缓存加速模块

##### 1. xcache缓存加速插件说明

XCache是一个开源的、又快有稳定的PHP opcode缓存器/优化器，其项目leader曾经是Lighttpd（和nginx类似的告诉web服务软件）的开发成员之一。XCache通过把对PHP程序编译后的数据（opcode）缓存到共享内存里，避免让相同的程序重复编译。用户请求相同的PHP程序时，可以直接使用缓存中已编译好的数据，从而提高PHP的访问速度，通常可以提升2~5倍，并大幅降低服务器负载开销。

很多公司在使用XCache，它已经能在大流量/高负载的成产环境稳定运行，与同类型的opcode缓存器相比在各个方面都更胜一筹，例如：社区活跃、快速开发、能够快速跟进PHP的版本更新等。

当前稳定版本为3.1.x（全面支持PHP5.1~5.5）和3.2.x（2014年底发布，全面支持PHP5.1~5.6）。

有关XCache软件想请请参考：

<http://xcache.lighttpd.net>或<http://xcache.lighttpd.net/wiki/introduction>

##### 2. XCache插件的安装过程

本章选择XCache最新稳定版本3.2.x（全面支持PHP5.1~5.6）为例进行讲解，其安装步骤及过程如下。

具体的安装目录集如下：

[root@lnmp tools]# cd /home/oldboy/tools/

[root@lnmp tools]# wget http://xcache.lighttpd.net/pub/Releases/3.2.0/xcache-3.2.0.tar.bz2

[root@lnmp tools]# tar xf xcache-3.2.0.tar.bz2

[root@lnmp tools]# cd xcache-3.2.0

[root@lnmp xcache-3.2.0]# /application/php/bin/phpize

[root@lnmp xcache-3.2.0]# ./configure --enable-xcache --with-php-config=/application/php/bin/php-config

[root@lnmp xcache-3.2.0]# make && make install

[root@lnmp xcache-3.2.0]# ls -l /application/php5.3.27/lib/php/extensions/no-debug-non-zts-20090626/

total 1052

-rwxr-xr-x 1 root root 417069 Jan 19 10:49 eaccelerator.so

-rwxr-xr-x 1 root root 658468 Jan 19 11:31 xcache.so ##最后生成了xcache.so模块就表示xcache安装成功。

[root@lnmp xcache-3.2.0]# cd ..

### 1.2.3 APC缓存及加速插件说明

APC全称为alternative PHP cache，也是一款不错的PHP缓存加速软件，限于篇幅，本节主讲xcache、eaccelerator、zendopcache，未对APC加速软件做深入讲解，仅仅在此处提及，如果对APC很感兴趣，可以参阅相关资料文档。

相关资料：

<http://pecl.php.net/packages.php>

<http://pecl.php.net/package/APC>

### 1.2.4 PHP官方插件zendopcache

##### 1. zendopcache插件说明

上面讲解了目前常见的PHP缓存加速插件：APC、eaccelerator、xcache，从PHP5.5开始，官方已经集成了新一代的缓存加速插件，名字为zendopcache，功能和前三者相似但又有少许不同，据官方概述，这个zendopcache缓存速度更快。

这几个PHP加速插件的主要原理基本相同，就是把PHP执行后的数据缓存到内存中从而避免重复的编译过程，使其能够直接使用缓存中已编译的代码，从而提高速度，降低服务器负载，他们的效率是显而易见的，想一些大型的CMS，每次打开一个页面要调用数十个PHP文件，执行数万行代码，效率可想而知，在安装上述加速器后打开页面的速度明显加快。

PHP5.5以上版本，支持zendopcache很简单，只需在编译PHP5.5的时候加上—enable-opcache就行了。其实，在PHP5.5版本以前，zendopcache也有独立的软件，并且也支持低版本的PHP5.2.\*，5.3.\*，5.4.\*，下面就以PHP5.3版本为例讲解zendopcache软件，以PHP扩展插件的方式介绍安装步骤。

官方下载地址为：<http://pecl.php.net/package/ZendOpcache>.

##### 2. zendopcache插件安装过程

这里选择以前的最新稳定版本：zendopcache-7.0.5.tgz为例进行介绍，由于使用的PHP版本为5.3，因此需要一PHP扩展的插件的方式安装，而不能使用PHP编译直接加参数（--enable-opcache）的方式（PHP5.5以上才可以），操作步骤及过程如下。

具体的安装命令集如下：

[root@lnmp tools]# cd /home/oldboy/tools/

[root@lnmp tools]# wget -q http://pecl.php.net/get/zendopcache-7.0.5.tgz

[root@lnmp tools]# tar xf zendopcache-7.0.5.tgz

[root@lnmp tools]# cd zendopcache-7.0.5

[root@lnmp zendopcache-7.0.5]# /application/php/bin/phpize

[root@lnmp zendopcache-7.0.5]# ./configure --enable-opcache --with-php-config=/application/php/bin/php-config

[root@lnmp zendopcache-7.0.5]# make && make install

[root@lnmp zendopcache-7.0.5]# ll /application/php5.3.27/lib/php/extensions/no-debug-non-zts-20090626/

total 1536

-rwxr-xr-x 1 root root 417069 Jan 19 10:49 eaccelerator.so

-rwxr-xr-x 1 root root 491814 Jan 19 11:48 opcache.so ##最后生成了opcache.so模块就表示opcache安装成功。

-rwxr-xr-x 1 root root 658468 Jan 19 11:31 xcache.so

[root@lnmp zendopcache-7.0.5]# cd ..

## 1.3 安装数据库缓存及其他PHP扩展插件

### 1.3.1 安装PHP memcached扩展插件

##### 1. memcached缓存软件说明

memcached是什么？

memcached是一个开源的、支持高性能、高并发以及分布式的内存缓存服务软件，从名称上看，前3个字符的单词mem就是内存的意思，而后面的5个字符的单词cache就是缓存的意思，最后的字符d，是daemon的意思，代表是服务端进程模式服务。

memcached服务分为服务端和客户端两部分，期中，服务端软件的名字形如memcached-1.4.13.tar.gz，客户端软件的名字形如memcached-2.27.tar.gz。

memcached诞生于2003年，最初由livejournal的Brad Fitzpatrick开发完成。memcached是整个项目的名称，而memcached是服务器端的主程序名，因其协议简单，且支持高并发而别广泛使用。

memcached的作用

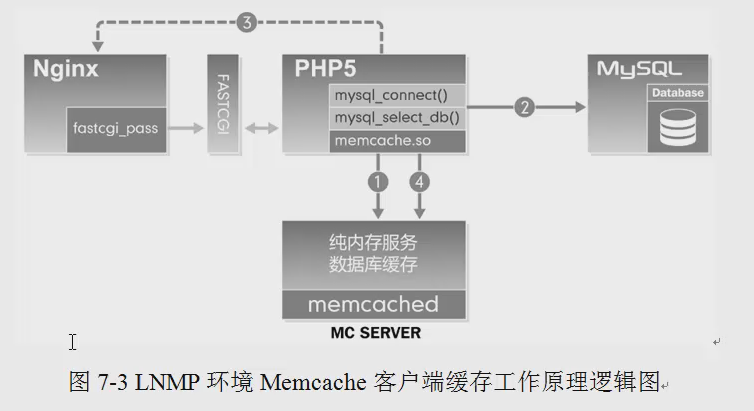
传统场景下，多数web应用都将数据保存到RDBMS中，www服务器从中读取数据并在浏览器中显示。但随着数据量的增大、访问的集中，就会出现RDBMS的负担加重、数据库响应缓慢、网站打开延迟等恶劣影响。

这时就需要memcached出马了。memcached是高性能的分布式内存缓存服务。使用memcached的主要目的是，通过在自身内存中缓存数据库的查询结果，减少数据库访问次数，以提高动态web应用的速度、提高网站架构的并发能力和可扩展性。

memcached服务通过在事先规划的系统内存空间中临时缓存数据库中的各类数据，以达到减少前端业务对数据库的直接高并发访问，从而提升大规模网站集群中动态服务的并发访问能力。

生产场景的memcached服务一般被用来保存网站中经常被读取的对象或数据，就想我们的客户端浏览器也会把经常访问的网页缓存起来一样，通过内存缓存来存取对象或数据要比磁盘存取快很多，因为磁盘是机械的介质，因此，在当今的IT企业中，memcached的应用范围很广。

memcached缓存架构应用场景逻辑图，下图是memcached缓存架构逻辑图。



具体步骤如下：

第一步：程序首先检查客户端请求的数据是否在memcached服务的缓存中存在，如果存在，直接把请求的数据返回，不在请求后端数据库。

第二步：如果请求的数据在memcached缓存中不存在，则程序会去memcached后端的数据库服务。

第三步：把从数据库中渠道的数据返回给客户端。

第四步：同时把新取到的数据库的数据缓存一份到memcached服务缓存中，下次同样的请求就直接从memcached服务缓存返回数据，从而减轻了数据库的访问压力。

##### 2. memcached缓存PHP扩展插件安装

前文已经提过，memcached分为服务端软件和客户端插件两部分，本文是memcached客户端PHP的扩展插件（memcached-2.2.7.tgz）在PHP环境中的安装，用于访问memcached服务端数据。

PHP的memcached扩展插件下载地址为：<http://pecl.php.net/package/memcache>.

PHP的memcached客户端扩展插件安装命令集如下：

[root@lnmp tools]# cd /home/oldboy/tools/

[root@lnmp tools]# wget -q http://pecl.php.net/get/memcache-2.2.7.tgz

[root@lnmp tools]# tar zxf memcache-2.2.7.tgz

[root@lnmp tools]# cd memcache-2.2.7

[root@lnmp memcache-2.2.7]# /application/php/bin/phpize

[root@lnmp memcache-2.2.7]# ./configure --enable-memcache --with-php-config=/application/php/bin/php-config

[root@lnmp memcache-2.2.7]# make && make install

[root@lnmp memcache-2.2.7]# cd ../

[root@lnmp tools]# ll /application/php5.3.27/lib/php/extensions/no-debug-non-zts-20090626/

total 1780

-rwxr-xr-x 1 root root 417069 Jan 19 10:49 eaccelerator.so

-rwxr-xr-x 1 root root 246752 Jan 19 12:07 memcache.so ##最后生成了memcache.so模块就表示memcache扩展插件成功安装。

-rwxr-xr-x 1 root root 491814 Jan 19 11:48 opcache.so

-rwxr-xr-x 1 root root 658468 Jan 19 11:31 xcache.so

提示：如果安装的是memcached-2.2.4.tgz，可能会报如下错误：make：\*\*\* [memcache.lo] Error1.

解决方法为使用如下命令：cp memcache.loT memcache.lo

##### 拓展：redis安装

为php安装redis客户端扩展  
获取源码包：

[root@mysql01 redis-3.0.6]# cd /home/yongzhen/tools/  
[root@mysql01 tools]# wget https://github.com/nicolasff/phpredis/archive/master.zip   
  
[root@mysql01 tools]# ll master.zip   
-rw-r--r-- 1 root root 149905 Jan 13 15:09 master.zip  
安装：  
[root@mysql01 tools]# unzip master.zip  
[root@mysql01 tools]# cd phpredis-master/  
[root@mysql01 phpredis-master]# /application/php/bin/phpize  
Configuring for:  
PHP Api Version:         20090626  
Zend Module Api No:      20090626  
Zend Extension Api No:   220090626  
[root@mysql01 phpredis-master]# ./configure -with-php-config=/application/php/bin/php-config  
[root@mysql01 phpredis-master]# make && make install  
修改php.ini设置，重启php：  
[root@mysql01 phpredis-master]# echo "extension = redis.so">>/application/php/lib/php.ini  
网页访问：http://www.etiantian.org/phpinfo.php

### 1.3.2 安装PDO\_MYSQL扩展模块

##### 1. PDO\_MYSQL 扩展插件说明

PDO扩展为PHP访问数据库定义了一个轻量级一致性的接口，它提供了一个数据访问抽象层，这样，无论使用的是什么数据库，都可以通过一致的函数执行查询并获取数据。PDO\_MSYQL扩展插件下载地址：<http://pecl.php.net/get/PDO_MYSQL-1.0.2.tgz>。

##### 2. PDO\_MYSQL扩展插件的安装过程

PDO\_MYSQL的安装有两种方法，一种插件方式安装，一种是编译PHP时加入PDO\_MYSQL支持，直接指定PHP的对应PDO\_MYSQL编译参数安装即可，例如：--with-pdo-mysql=mysqlnd，同时PHP的环境也可以不装MySQL软件，直接指定如下参数—with-mysql=mysqlnd即可让PHP支持连接MySQL数据库。当然了，建议你跟着本章节的演示进行，先把路走通了，再去测试提到的方法。此处选择工作中常用的PHP扩展插件方式安装。

完整安装PDO\_MYSQL命令集如下：

[root@lnmp ~]# cd /home/oldboy/tools/

[root@lnmp tools]# wget -q <http://pecl.php.net/get/PDO_MYSQL-1.0.2.tgz>

[root@lnmp tools]# tar xf PDO\_MYSQL-1.0.2.tgz

[root@lnmp tools]# cd PDO\_MYSQL-1.0.2

[root@lnmp PDO\_MYSQL-1.0.2]# /application/php/bin/phpize

[root@lnmp PDO\_MYSQL-1.0.2]# ./configure --with-php-config=/application/php/bin/php-config --with-pdo-mysql=/application/mysql

##configure的参数路径要正确配置。/application/mysql为MySQL的安装路径，要先安装好MySQL。

[root@lnmp PDO\_MYSQL-1.0.2]# ll /application/php5.3.27/lib/php/extensions/no-debug-non-zts-20090626/

total 1936

-rwxr-xr-x 1 root root 417069 Jan 19 10:49 eaccelerator.so

-rwxr-xr-x 1 root root 246752 Jan 19 12:07 memcache.so

-rwxr-xr-x 1 root root 491814 Jan 19 11:48 opcache.so

-rwxr-xr-x 1 root root 156004 Jan 19 13:33 pdo\_mysql.so ##出现pdo\_mysql.so模块表示pdo\_mysql成功安装。

-rwxr-xr-x 1 root root 658468 Jan 19 11:31 xcache.so

## 1.4 安装其他的PHP扩展插件模块

### 1.4.1 安装图像处理程序及imagick扩展模块

##### 1. 安装ImageMagick图像软件

ImageMagick是一套功能强大、稳定而且免费的工具集合开发包，可以用来读、写和处理超过89种基本格式的图片文件，包括流行的TIFF、JPEG、GIF、PNG、PDF以及photoCD等。利用ImageMagick，可以根据web应用程序的需要动态生成图片，还可以对一个（或一组）图片进行改变大小、旋转、锐化、减色或增加特效等操作，并将操作的结果以相同格式或其他格式保存，对图片的操作，即可以通过命令进行，也可以用C/C++、Perl、Java、PHP、Python或ruby编程来完成。同时ImageMagick提供了一个高质量的2D工具包，部分支持SVG。现在，ImageMagick的主要精力集中在加强性能、减少bug以及提供稳定的API和ABI上。

ImageMagick的常见功能如下：

* 将图片从一个格式转换到另一个格式，包括直接转换成图标。
* 可以改变图片尺寸、旋转、锐化（sharpen）、减色、设置图片特效。
* 对图片设置各种尺寸缩略图。
* 可以适应于web背景的透明图片。
* 将一组图片做成GIF动画，直接convert。
* 将几张图片做成一张组合图片。
* 在一个图片上写字或画图形，带文字阴影和边框渲染。
* 给图片加边框或框架。
* 取得一些图片的特性信息。

它几乎包括了gimp可以实现的所有常规插件功能，甚至包括各种曲线参数的渲染功能。ImageMagick的下载地址：<http://download.chinaunix.net/download/0001000/95.shtml>，请提前下载好放到指定服务器的目录下。

安装ImageMagick的命令集如下：

[root@lnmp tools]# ls -l ImageMagick-6.7.9-9.tar.xz

-rw-r--r-- 1 root root 8363720 Sep 7 11:52 ImageMagick-6.7.9-9.tar.xz

[root@lnmp tools]# tar xf ImageMagick-6.7.9-9.tar.xz

[root@lnmp tools]# cd ImageMagick-6.7.9-9

[root@lnmp ImageMagick-6.7.9-9]# ./configure

[root@lnmp ImageMagick-6.7.9-9]# make && make install

[root@lnmp ImageMagick-6.7.9-9]# cd ../

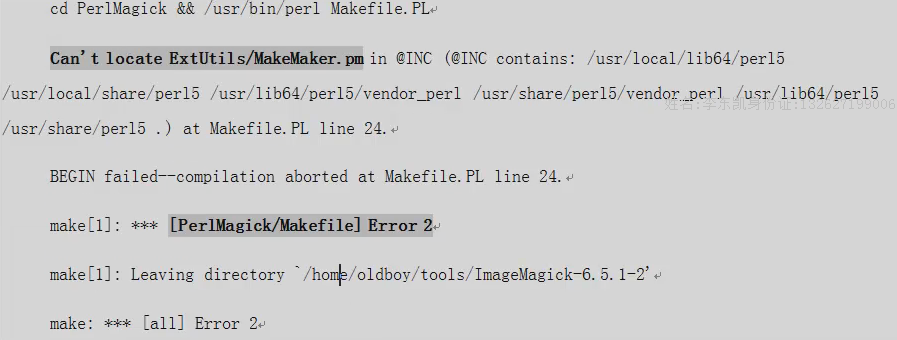
#提示：此图像软件安装时make步骤耗时较长！

#提示：此步不是安装PHP的扩展，因此，没有生产.so的文件。

下面来看看ImageMagick安装报错及解决方法。

问题1：make步骤出错。

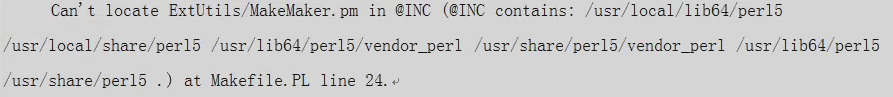
示例如下：



解决方法：采用 yum install perl-devel --y 命令。

解决思路：

方法一，根据如下内容：

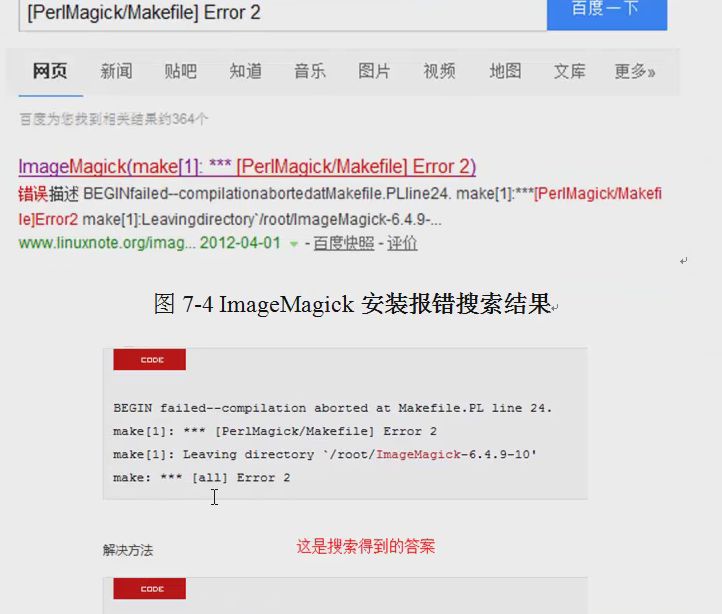


可以看到，上述内容有Makefile.PL、/usr/lib64/perl5/vendor\_perl和Perl语言的字样，因此可以试着使用yum install --y perl-devel命令安装相关包看看是否可以搞好。前文讲解过PHP基础依赖包的安装方法，遇到问题首先要考虑装软件依赖的devel包，多去思考水平自然就提高了，哪怕思考的不对也锻炼思维了，次数多了准确率就高了。

方法二，寻找唯一并且有特征的错误提示去搜索，如下：

[PerMagick/Makefile] Error 2

下图是通过搜索得到的内容，利用搜索引擎快速搜索需要的信息，是运维人员重要的能力，一般报错问题都可以搜索解决，尤其是能谷歌搜索最好。



问题2：报错：gcc:internal compiler error:Killed(program cc1)

解决方法：通过查看dmesg发现下述错误信息



原因:VM虚拟机安装时给的内存小了（400MB太小，大于512MB就没事），虚拟机安装应该考虑1GB加的VM内存较好。

##### 2. 安装imagick php扩展插件

imagick插件工作㤇ImageMagick软件的支持，所以，必须要先安装ImageMagick，否则会报错，见上文。

imagick插件式一个可以供PHP调用ImageMagick功能的扩展模块。使用这个扩展可以使PHP具备和ImageMagick相同的功能。

安装了ImageMagick图片的创建与修改、压缩等操作，因为它们都集成在了imagick这个PHP扩展中。

其安装命令集如下（需要提前下载好放到指定服务器目录下）：

[root@lnmp tools]# tar zxf imagick-2.3.0.tgz

[root@lnmp tools]# cd imagick-2.3.0

[root@lnmp imagick-2.3.0]# /application/php/bin/phpize

[root@lnmp imagick-2.3.0]# ./configure --with-php-config=/application/php/bin/php-config

##configure的参数路径要正确配置。

[root@lnmp imagick-2.3.0]# make && make install

[root@lnmp imagick-2.3.0]# ll /application/php5.3.27/lib/php/extensions/no-debug-non-zts-20090626/

total 2984

-rwxr-xr-x 1 root root 417069 Jan 19 10:49 eaccelerator.so

-rwxr-xr-x 1 root root 1073033 Jan 19 14:23 imagick.so ##最后生成了imagick.so模块就对了。

-rwxr-xr-x 1 root root 246752 Jan 19 12:07 memcache.so

-rwxr-xr-x 1 root root 491814 Jan 19 11:48 opcache.so

-rwxr-xr-x 1 root root 156004 Jan 19 13:33 pdo\_mysql.so

-rwxr-xr-x 1 root root 658468 Jan 19 11:31 xcache.so

[root@lnmp imagick-2.3.0]# cd ..

### 1.4.2 检查所有PHP扩展插件模块安装的成果

到此为止，常见的PHP扩展插件安装得就差不多了，下面看看我们安装的成果吧。

[root@lnmp tools]# ll /application/php5.3.27/lib/php/extensions/no-debug-non-zts-20090626/

total 2984

-rwxr-xr-x 1 root root 417069 Jan 19 10:49 eaccelerator.so

-rwxr-xr-x 1 root root 1073033 Jan 19 14:23 imagick.so

-rwxr-xr-x 1 root root 246752 Jan 19 12:07 memcache.so

-rwxr-xr-x 1 root root 491814 Jan 19 11:48 opcache.so

-rwxr-xr-x 1 root root 156004 Jan 19 13:33 pdo\_mysql.so

-rwxr-xr-x 1 root root 658468 Jan 19 11:31 xcache.so

当前一共6个常用扩展模块，其他需要的以后可以后需安装。其中，eaccelerator.so、opcache.so、xcache.so属于同类软件，生产环境中安装其中一种即可，否则，可能会引起同时使用冲突，这里全都介绍了，目的是让大家了解方法。另外，pdo\_mysql.so、imagick.so属于功能软件，可选安装，memcache.so是数据库缓存软件，可选安装。

## 1.5 配置PHP加速与缓存相关扩展插件模块

### 1.5.1 配置memcache/pdo\_mysql/imagick模块生效

##### 1.修改PHP的配置文件过程如下：

1)执行vi/application/php/lib/php.ini命令，编辑查找extension\_dir=”./”参数，修改为extension\_dir=”/pplication/php5.3.27/lib/php/extensions/no-debug-non-zts-20090626”，这个extension\_dir对应的路径就是前文编译的模块所在的路径。

提示：默认的php配置文件路径为/application/php/lib/php.ini，可以通过在编译php时添加参数指定php.ini的配置路径--with-config-file-path=/application/php5.3.27/lib/etc，如果不指定编译路径，默认为/application/php/lib。

快速操作命令的方法如下：

[root@lnmp imagick-2.3.0]# cd /application/php/lib/

[root@lnmp lib]# cp php.ini php.ini.ori

-rw-r--r-- 1 root root 69627 Jan 19 16:13 php.ini.ori ##修改前备份

vim php.ini

在最后一行开始编辑，添加如下几行信息：

extension\_dir = "/application/php5.3.27/lib/php/extensions/no-debug-non-zts-20090626/"

extension = memcache.so

extension = pdo\_mysql.so

extension = imagick.so

提示：注释不是“#”，而是“;”

2.检查配置id相关模块生效情况

（1） 重启php服务，编写测试程序phpinfo

首先，重启php服务，然后检查模块生效情况，命令如下：

[root@lnmp lib]# ps -ef|grep php-fpm

root 1348 1 0 21:38 ? 00:00:00 php-fpm: master process (/application/php5.3.27/etc/php-fpm.conf)

nginx 1349 1348 0 21:38 ? 00:00:00 php-fpm: pool www

nginx 1350 1348 0 21:38 ? 00:00:00 php-fpm: pool www

root 46444 1742 0 22:23 pts/2 00:00:00 grep php-fpm

[root@lnmp lib]# pkill php-fpm

[root@lnmp lib]# ps -ef|grep php-fpm

root 46447 1742 0 22:23 pts/2 00:00:00 grep php-fpm

[root@lnmp lib]# /application/php/sbin/php-fpm

[root@lnmp lib]# ps -ef|grep php-fpm

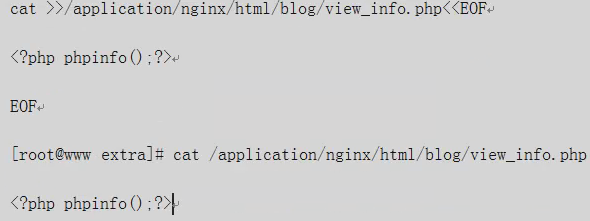
ot 46450 1 0 22:23 ? 00:00:00 php-fpm: master process (/application/php5.3.27/etc/php-fpm.conf)

nginx 46451 46450 0 22:23 ? 00:00:00 php-fpm: pool www

nginx 46452 46450 0 22:23 ? 00:00:00 php-fpm: pool www

root 46454 1742 0 22:23 pts/2 00:00:00 grep php-fpm

然后在前文提高的blog程序的站点目录下增加phpinfo.php代码文件：

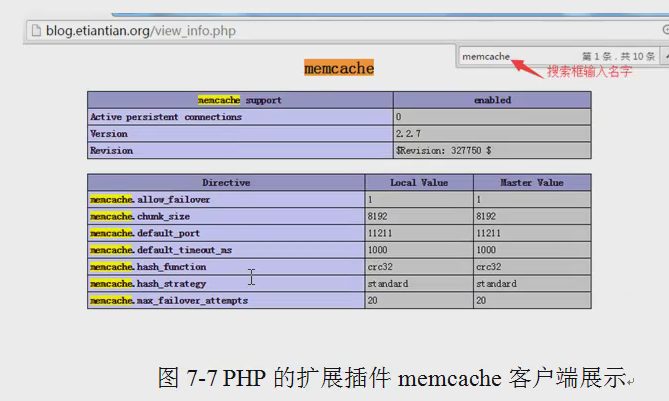


（2）检查memcached扩展插件

配好客户端的host解析，然后在浏览器中输入<http://blog.etiantian.org/phpinfo.php>页面的地址，出现的内容如图所示。

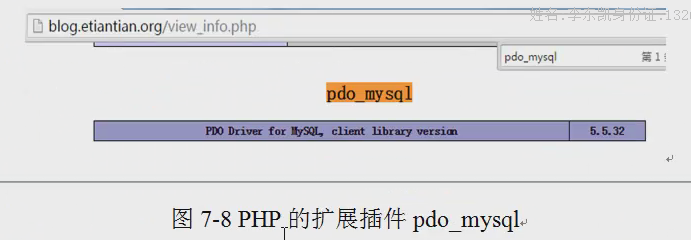


通过快捷键Ctrl+f进行页面搜索，如果找到图7-7所示的内容，表示memcached插件已生效。



（3）检查PD0\_mysql扩展插件

同理搜索检查PDO\_mysql扩展插件，如图7-8所示。



（4） 检查imagick扩展插件

同理搜索检查imagick扩展插件，如图7-9所示。

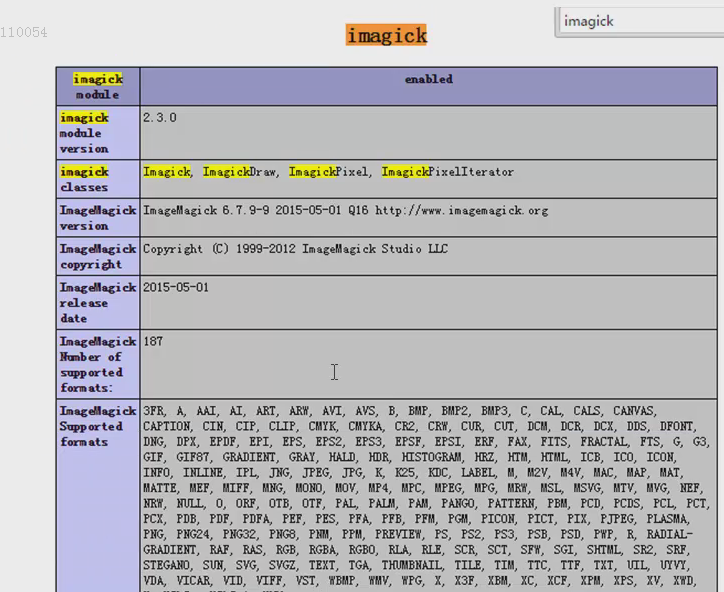


图7-9php的扩展插件imagick

到此为止，pdo\_mysql.so、imagick.so、memcache.so这三个PHP的扩展插件就全部安装及配置完毕，后面将配置其余的缓存插件。

### 1.5.2 配置eaccelerator插件生效并优化参数

##### 1. 配置eaccelerator缓存目录

配置命令1：配置eaccelerator缓存目录

mkdir -p /tmp/eaccelerator

##此目录可以用tmpfs内存文件系统或者SSD固态硬盘来存储。

chown -R nginx.nginx /tmp/eaccelerator

提示：上面chown后的用户是nginx的用户。

[root@lnmp lib]# ll -d /tmp/eaccelerator/

drwxr-xr-x 2 nginx nginx 4096 Jan 19 23:12 /tmp/eaccelerator/

配置命令2：配置eaccelerator参数：

cat >> /application/php/lib/php.ini<<EOF

[eaccelerator]

extension=eaccelerator.so

eaccelerator.shm\_size="64"

eaccelerator.cache\_dir="/tmp/eaccelerator"

eaccelerator.enable="1"

eaccelerator.optimizer="1"

eaccelerator.check\_mtime="1"

eaccelerator.debug="0"

eaccelerator.filter=""

eaccelerator.shm\_max="0"

eaccelerator.shm\_ttl="3600"

eaccelerator.shm\_prune\_period="3600"

eaccelerator.shm\_only="0"

eaccelerator.compress="1"

eaccelerator.compress\_level="9"

EOF

操作后检查配置结果

[root@lnmp blog]# tail -15 /application/php/lib/php.ini

[eaccelerator]

extension=eaccelerator.so

eaccelerator.shm\_size="64"

eaccelerator.cache\_dir="/tmp/eaccelerator"

eaccelerator.enable="1"

eaccelerator.optimizer="1"

eaccelerator.check\_mtime="1"

eaccelerator.debug="0"

eaccelerator.filter=""

eaccelerator.shm\_max="0"

eaccelerator.shm\_ttl="3600"

eaccelerator.shm\_prune\_period="3600"

eaccelerator.shm\_only="0"

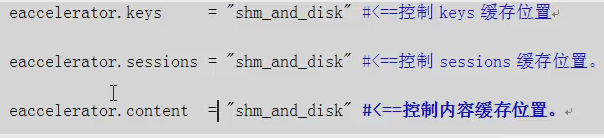
eaccelerator.compress="1"

eaccelerator.compress\_level="9"

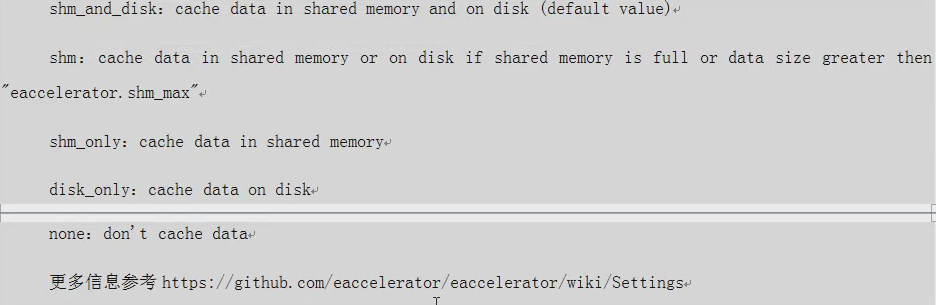
下表给出了eaccelerator配置参数的详细情况。

|  |  |
| --- | --- |
| **eaccelerator参数** | **解释说明** |
| [eaccelerator] | 开始eaccelerator加速模块配置 |
| extension=eaccelerator.so | 加载eaccelerator加速模块，路径相对于extension\_dir的配置。 |
| eaccelerator.shm\_size="64" | 存储缓存数据的共享内存大小，如果为0，则最大值看内核配置/proc/sys/kernel/shmmax。 |
| eaccelerator.cache\_dir="/tmp/eaccelerator" | 磁盘缓存存储路径，缓存内容为precompiled code,session data,content and user entries here. 缺省路径为“/tmp/eaccelerator”。 |
| eaccelerator.enable="1" | Eaccelerator缓存生效开关，1为默认值，即生效，0为关闭，不生效。 |
| eaccelerator.optimizer="1" | 加速PHP代码执行速度，1为默认值表示激活，0为不激活。用于缓存前的代码加速。 |
| eaccelerator.check\_mtime="1" | 检查缓存修改时间决定代码是否徐璈重新编译，1为激活，是默认值。 |
| eaccelerator.debug="0" | 缓存加速调试，0为关闭，1为打开，打开后可以看到缓存命中信息。 |
| eaccelerator.filter="" | 设定对象是否缓存规则，空表示不设定。 |
| eaccelerator.shm\_max="0" | 可以被放置的最大缓存最大大小，0是不限制。 |
| eaccelerator.shm\_ttl="3600" | 缓存文件的生存期。 |
| eaccelerator.shm\_prune\_period="3600" | 当共享内存空间不够时，从共享内存移除老数据的时间周期。 |
| eaccelerator.shm\_only="0" | 是否允许缓存数据到磁盘，0为允许，但是对于session data and content caching无影响。 |
| eaccelerator.compress="1" | 是否开启压缩，1为开启。 |
| eaccelerator.compress\_level="9" | 压缩级别，9为最高。 |

其他根据内容指定是否缓存到共享内存或磁盘参数：



上述3个参数可选的值为：



##### 2. 检查eaccelerator加速配置情况

提示：如果仅检查加速情况配置，可以不重启Apache。

执行PHP命令，测试缓存的配置情况。

[root@lnmp blog]# /application/php/bin/php -v

PHP 5.3.27 (cli) (built: Dec 4 2015 13:23:55)

Copyright (c) 1997-2013 The PHP Group

Zend Engine v2.3.0, Copyright (c) 1998-2013 Zend Technologies

with eAccelerator v0.9.6.1, Copyright (c) 2004-2010 eAccelerator, by eAccelerator

重启PHP服务后才可生效：

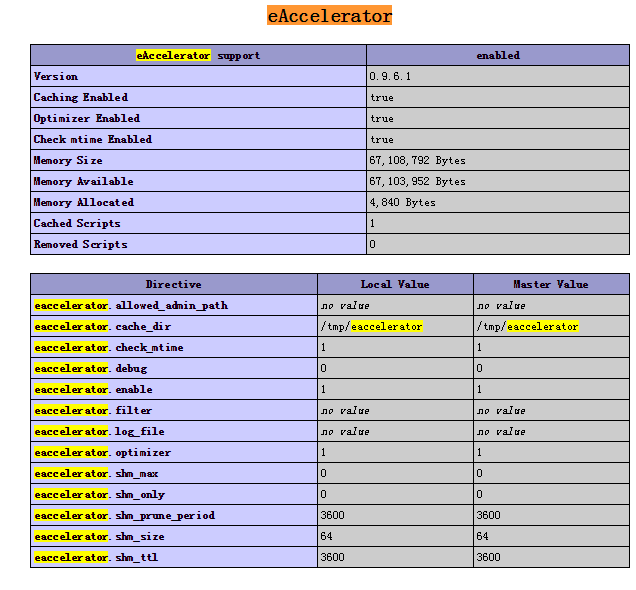
[root@lnmp blog]# pkill php-fpm

[root@lnmp blog]# ps -ef|grep php-fpm

root 46986 46743 0 10:17 pts/0 00:00:00 grep php-fpm

[root@lnmp blog]# /application/php/sbin/php-fpm

现在通过phpinfo检查eaccelerator插件结果，如图所示：



此时看看缓存目录/tmp/eaccelerator：

[root@lnmp blog]# ls -l /tmp/eaccelerator/

total 64

drwxrwxrwx 18 root root 4096 Jan 20 10:08 0

drwxrwxrwx 18 root root 4096 Jan 20 10:08 1

drwxrwxrwx 18 root root 4096 Jan 20 10:08 2

drwxrwxrwx 18 root root 4096 Jan 20 10:08 3

drwxrwxrwx 18 root root 4096 Jan 20 10:08 4

drwxrwxrwx 18 root root 4096 Jan 20 10:08 5

drwxrwxrwx 18 root root 4096 Jan 20 10:08 6

drwxrwxrwx 18 root root 4096 Jan 20 10:08 7

drwxrwxrwx 18 root root 4096 Jan 20 10:08 8

drwxrwxrwx 18 root root 4096 Jan 20 10:08 9

drwxrwxrwx 18 root root 4096 Jan 20 10:08 a

drwxrwxrwx 18 root root 4096 Jan 20 10:08 b

drwxrwxrwx 18 root root 4096 Jan 20 10:08 c

drwxrwxrwx 18 root root 4096 Jan 20 10:08 d

drwxrwxrwx 18 root root 4096 Jan 20 10:08 e

drwxrwxrwx 18 root root 4096 Jan 20 10:08

可以看到/tmp/eaccelerator/缓存目录下也有内容了，以上两个检查可以确保配置生效了。

##### 3. 访问PHP页面测试检查eaccelerator加速情况。

重启PHP服务后，子浏览器里访问PHP页面，如phpinfo页面，又会有下面的缓存文件，其实上面已经访问过了。

[root@lnmp blog]# find /tmp/eaccelerator/ -type f|xargs file

/tmp/eaccelerator/8/a/eaccelerator-1502.507103: data

/tmp/eaccelerator/8/a/eaccelerator-1502.017103: data

/tmp/eaccelerator/8/a/eaccelerator-1502.896103: data

/tmp/eaccelerator/8/a/eaccelerator-1502.407103: data

/tmp/eaccelerator/8/f/eaccelerator-1502.082104: data

/tmp/eaccelerator/3/b/eaccelerator-1502.378103: data

/tmp/eaccelerator/3/b/eaccelerator-1502.278103: data

提示：eaccelerator-1502.507103就是cache的内容，而且是phpinfo的页面缓存内容，类型为data。

##### 4． 使用tmpfs优化eaccelerator缓存目录

tmpfs是一种基于内存的文件系统，使用tmpfs作为数据临时存储通常比本地磁盘快很多，此方法适用于临时使用的各类缓存场景。例如：上传图片时很多软件默认在/tmp下临时缓存切图、存放session数据，则可以让/tmp使用tmpfs文件系统来加快访问效率，本文将/tmp/eaccelerator挂载到tmpfs文件系统上，让访问缓存的数据更快。具体操作方法为：

[root@lnmp blog]# mount -t tmpfs -o size=16m tmpfs /tmp/eaccelerator/

##创建16M大小的tmpfs类型文件系统挂载到/tmp/eaccelerator

[root@lnmp blog]# df --h ###检查挂载情况

Filesystem Size Used Avail Use% Mounted on

/dev/sda3 6.9G 4.1G 2.5G 63% /

tmpfs 932M 0 932M 0% /dev/shm

/dev/sda1 190M 36M 145M 20% /boot

tmpfs 16M 0 16M 0% /tmp/eaccelerator

[root@lnmp blog]# grep eacc /proc/mounts ##检查挂载情况

tmpfs /tmp/eaccelerator tmpfs rw,relatime,size=16384k 0 0

### 1.5.3 配置xcache插件加速

特别提示：xcache和eaccelerator功能相近，安装一个即可。本章节考虑知识完整性，作为知识点来讲解，配置之前应将eaccelerator的所有配置删除。

##### 1. xcache软件的解压目录/home/oldboy/tools/xcache-3.2.0/下存在一个名字为xcache.ini的配置文件，即为xcache的配置文件。

下面的表给出了xcache配置文件参数的说明。

|  |  |
| --- | --- |
| **Xcache配置文件参数** | **解释说明** |
| [xcache-common]  extension = xcache.so | 加载xcache.so路径相对于extension\_dir的配置。自3.0版本开始不再支持使用zend\_extension加载xcache的方式。\*\*\*\*\* |
| [xcache.admin]  xcache.admin.enable\_auto = On | 激活管理员认证。 |
| xcache.admin.user = “mOo”  xcache.admin.pass = “md5 encrypted password” | 指定xcache管理员用户名和密码，密码根据  <http://xcache.lighttpd.net/demo/cacher/mkpassword.php>  地址产生，留空表示禁止管理页面。 |
| [xcache] | 开始xcache缓存参数配置段，下面所有的初始值即为默认值，除非明确说明。 |
| xcache.shm\_scheme = “mmap” | 设置xcache如何从系统分配共享内存。 |
| xcache.size = 60M | 0为禁止缓存，非0则启用缓存。需要注意系统所允许的mmap最大值。\*\*\*\*\* |
| xcache.conut = 1 | 指定将cache切分成多少块，官方推荐设置为服务器CPU的数量\*\*\*\*\*  [root@lnmp xcache-3.2.0]# grep -c proccessor /proc/cpuinfo  1 |
| xcache.slots = 8k | 只是作为hash槽个数的参考值，缓冲超过这个个数的内容没有任何问题（you can always store count(items)>slot） |
| xcache.ttl = 0 | 设定cache对象的生存期TTL（time to live），0为永不过期。\*\*\*\*\* |
| xcache.go\_interval = 0 | 回收器扫描过期的对象回收内存空间的间隔，0为不扫描，其他值的单位是秒。 |
| xcache.var\_size = 4M  xcache.var\_count = 1  xcache.var\_slots = 8k  xcache.var\_ttl = 0  xcache.var\_gc\_interval = 300 | 这三个值和上面的几个类似，只不过用于变量缓存，而不是opcode缓存。 |
| ;N/A for /dev/zero  xcache.readonly\_protection = off | 如果启用了该参数，将会略微降低性能，但会提高一定的安全系数。这个选项对于xcache.mmap\_path=/dev/zero无效。 |
| xcache.mmap\_path = “/dev/zero” | 对于\*nix,xcache.mmap\_path是一个文件路径而非目录。如果要启用该参数，请使用”/tmp/xcache”这样的路径，而不是”/dev/\*”，如果开启了xcache.readonly\_protection参数，不同进程组的PHP将不会共享同一个/tmp/xcache路径。 |
| xcache.coredump\_directory = “” | 当xcache crash后，是否把数据保存到指定路径。 |
| xcache.disable\_on\_crash = off | 当xcache发生crash时，自动关闭xcache缓存。 |

提示：有关更多参数说明请看配置文件对应注释说明或参考官方文档：

http://xcache.lighttpd.net/wiki/XcacheIni.

##### 2. 修改php.ini配置xcache

（1） 先在配置xcache参数前加个配置分界符，配置命令如下：

cd /application/php/lib

echo >>php.ini

echo ';xcache config by ll 20160120--------------' >>php.ini

tail -2 php.ini

执行过程如下：

[root@lnmp lib]# cd /application/php/lib

[root@lnmp lib]# echo >>php.ini

[root@lnmp lib]# echo ';xcache config by ll 20160120--------------' >>php.ini

[root@lnmp lib]# tail -2 php.ini

;xcache config by ll 20160120--------------

（2） 执行vim php.ini编辑修改xcache的配置参数，调整关键的参数见下表：

|  |  |
| --- | --- |
| **需要调整的关键参数** | **调整后的关键参数数值** |
| xcache.size = 60M  xcache.count = 1  xcache.ttl = 0  xcache.gc\_interval = 0  xcache.var\_size = 4M | xcache.size = 256M  xcache.count = 2  xcache.ttl = 86400  xcache.gc\_interval = 3600  xcache.var\_size = 0 |

以上参数需要根据生产硬件的大小，以及业务数据的访问量来调整，修改相应配置参数后保存退出。

（3） 强修改后的xcache.ini合并到php.ini结尾。命令如下：

[root@lnmp lib]# cat /home/oldboy/tools/xcache-3.2.0/xcache.ini >>php.ini

整个xcache.ini内容修改后的配置为：

[root@lnmp lib]# tail -85 php.ini

;xcache config by ll 20160120--------------

;; this is an example, it won't work unless properly configured into php.ini

[xcache-common]

;; non-Windows example:

extension = xcache.so

;; Windows example:

; extension = php\_xcache.dll

[xcache.admin]

xcache.admin.enable\_auth = On

......

执行完以上操作后，重启php会报错，因为系统不兼容连个软件同时公用：

[20-Jan-2016 11:38:19] NOTICE: PHP message: PHP Warning: Cannot load module 'XCache' because conflicting module 'eAccelerator' is already loaded in Unknown on line 0

[root@lnmp lib]# /application/php/sbin/php-fpm -v

[20-Jan-2016 11:39:02] NOTICE: PHP message: PHP Warning: Cannot load module 'XCache' because conflicting module 'eAccelerator' is already loaded in Unknown on line 0

PHP 5.3.27 (fpm-fcgi) (built: Dec 4 2015 13:23:43)

Copyright (c) 1997-2013 The PHP Group

Zend Engine v2.3.0, Copyright (c) 1998-2013 Zend Technologies

with eAccelerator v0.9.6.1, Copyright (c) 2004-2010 eAccelerator, by eAccelerator

将php.ini配置文件中的eaccelerator模块全部删除后即可生效。

[root@lnmp lib]# /application/php/sbin/php-fpm

[root@lnmp lib]# /application/php/sbin/php-fpm -v

PHP 5.3.27 (fpm-fcgi) (built: Dec 4 2015 13:23:43)

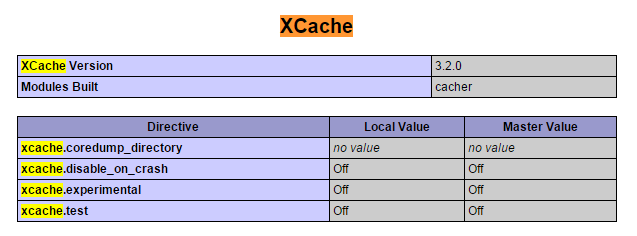
Copyright (c) 1997-2013 The PHP Group

Zend Engine v2.3.0, Copyright (c) 1998-2013 Zend Technologies

with XCache v3.2.0, Copyright (c) 2005-2014, by mOo

with XCache Cacher v3.2.0, Copyright (c) 2005-2014, by mOo

现在重启PHP服务，然后通过phpinfo界面检查xcache插件结果，如图所示：



### 1.5.4 配置zendopcache插件加速

##### 1. 配置zendopcache参数

在php.ini的最后面加入下面几行：

cat >> /application/php/lib/php.ini << EOF

[opcache]

zend\_extension=/application/php5.3.27/lib/php/extensions/no-debug-non-zts-20090626/opcache.so

opcache.memory\_consumption=128

opcache.interned\_strings\_buffer=8

opcache.max\_accelerated\_files=4000

opcache.revalidate\_freq=60

opcache.fast\_shutdown=1

opcache.enable\_cli=1

EOF

检查配置文件，查看配置结果：

[root@lnmp zendopcache-7.0.5]# tail /application/php/lib/php.ini

; make sure it's readable (open\_basedir is checked) by coverage viewer script

xcache.coveragedump\_directory = ""

[opcache]

zend\_extension=/application/php5.3.27/lib/php/extensions/no-debug-non-zts-20090626/opcache.so

opcache.memory\_consumption=128

opcache.interned\_strings\_buffer=8

opcache.max\_accelerated\_files=4000

opcache.revalidate\_freq=60

opcache.fast\_shutdown=1

opcache.enable\_cli=1

确认和自己希望配置的是否一致，操作完检查是运维人员的一个优秀的习惯。

##### 2. 检查zendopcache生效情况

下面使用PHP命令检查生效结果：

[root@lnmp zendopcache-7.0.5]# /application/php/bin/php -v

PHP 5.3.27 (cli) (built: Dec 4 2015 13:23:55)

Copyright (c) 1997-2013 The PHP Group

Zend Engine v2.3.0, Copyright (c) 1998-2013 Zend Technologies

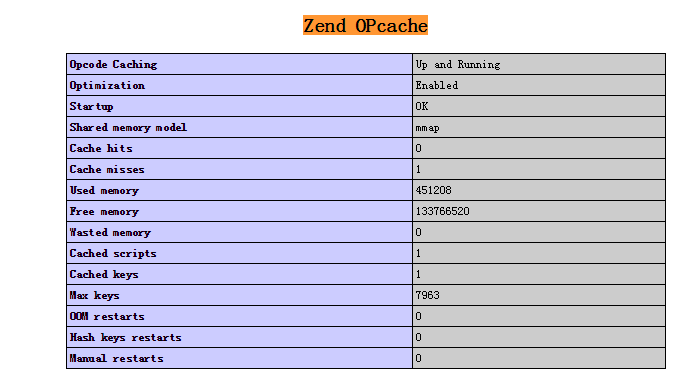
with XCache v3.2.0, Copyright (c) 2005-2014, by mOo

with Zend OPcache v7.0.5, Copyright (c) 1999-2015, by Zend Technologies

with XCache Cacher v3.2.0, Copyright (c) 2005-2014, by mOo

提示：可以看到zendopcache已经生效，并且貌似和xcache相处的比较融洽，不过工作中是否多选，还要慎重选择。

然后重启PHP服务，并通过phpinfo介苗检查xcache插件结果，如图下所示：



##### 3. zendopcache配置参数说明

下表中针对opcache的部分重要参数进行了说明

|  |  |
| --- | --- |
| **opcache参数** | **解释说明** |
| opcache.memory\_consumption = 128 | Opcache共享内存空间大小，用于存放precompiled PHP code，默认为64，单位为mbytes |
| opcache.interned\_strings\_buffer =8 | 默认值为4，interned strings内存的数量，单位是M |
| opcache.max\_accelerated\_files=4000 | 默认值为2000，opcache哈希表的key最大数量。 |
| opcache.revalidate\_freq=60 | 默认值为2，检查文件时间戳的频率，用于共享内存分配的变化。 |
| opcache.fast\_shutdown=1 | 默认值为0，如果激活，一个快速的关闭队列将被用来加速代码。 |
| opcache.enable\_cli=1 | 默认值为0，激活PHP CLI的opcache，用于测试和调试。 |

更多的opcache参数可以查看安装目录下的README

特别说明：

zendopcache是PHP官方的新一代的缓存加速软件，PHP5.5以前可以通过zendopcache软件以插件扩展的方式安装，PHP5.5版本开始已经整合到PHP软件里，编译时只需指定一个参数即可，例如：--enable-opcache。

zendopcache可能将是未来的首选，现在的稳定性还有待检验。小规模环境下，PHP5以上的版本可以使用，或者可以忍受其未知的问题也可以使用。

PHP5.5及以上版本PHP安装编译的完整参数为：

tar zxf php-5.5.13.tar.gz

cd php-5.5.13

./configure --prefix=/application/php --with-mysql --with-jpeg-dir --with-png-dir --with-zlib --enable-xml --with-libxml-dir --with-curl --enable-bcmath --enable-shmop --enable-sysvsem --enable-inline-optimization --enable-mbregex --with-openssl --enable-mbstring --with-gd --enable-gd-native-ttf --enable-sockets --with-xmlrpc --enable-zip --enable-soap --disable-debug --enable-opcache --enable-zip --enable-fpm --with-fpm-user=nginx --with-fpm-group=nginx

make && make install

在编译参数里指定--enable-opcache即可编译。

## 1.6 生产环境PHP扩展插件的安装建议

### 1.6.1 PHP的安装插件表格列表

|  |  |  |
| --- | --- | --- |
| **PHP EXT module** | **说明** | **备注** |
| eaccelerator-0.9.5.2.tar.tar | 适合PHP5.3以前的版本，PHP缓存加速 | 可选PHP扩展插件 |
| eaccelerator-0.9.6.tar.bz2 | 适合PHP5.3版本，PHP缓存加速 | 可选PHP扩展插件 |
| ImageMagick.tar.gz | 常用图像处理程序，属功能应用 | 非PHP的扩展插件 |
| imagick.2.3.0.tgz | 需要先装图像处理程序，属功能应用。 | 可选PHP扩展插件 |
| memcache-2.2.7.tgz | Memcache客户端数据库缓存优化用 | 可选PHP扩展插件 |
| PDO\_MYSQL-1.0.2.tgz | PHP数据库访问插件，属功能应用 | 可选PHP扩展插件 |
| xcache-3.2.0.tar.bz2 | 支持PHP5.1-5.6 PHP缓存加速 | 可选PHP扩展插件 |
| zendopcache-7.0.5.tgz | 支持PHP5.3-5.4，PHP缓存加速。 | 可选PHP扩展插件 |

### 1.6.2 生产环境插件的安装建议

1） 对于功能性插件，如果业务产品不需要使用，可以暂时不考虑安装，例如：PDO\_MYSQL\memcache\imagick等。如果不清楚是否需要，最好还是装上，有备无患。

2） 对于性能优化插件，eaccelerator、xcache、zendopcache、APC可以安装任意种，具体情况看实际业务需求，在选择时最好能搭建相关环境进行压力测试，然后根据实际测试结果来选择，用数据说话很重要。

建议一般的环境，上述4种优化插件都可以使用，效果都还可以。

### 1.6.3 PHP加速插件的测试对比

1. 压力测试结果列表

下表为相关PHP加速插件的测试结果对比参考（无opcache）

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
|  | 请求时间（秒） | 单次请求时间（毫秒） | 最大内存占用（MB） | 最小内存占用（MB） |
| None | 10.41 | 96.08 | 24 | 24 |
| APC | 30.45 | 32.84 | 21 | 21 |
| Eaccelerator | 31.26 | 31.99 | 23 | 18 |
| Xcache | 30.28 | 33.02 | 29 | 19 |

下面针对PHP加速器比较结果进行总结。

* 通过测试得出eaccelerator在请求时间和内存占用综合方面是最好的。
* 通过测试得出使用加速器比无加速器在请求时间快了3倍左右。
* 通过各个官方观察，xcache是更新最快的，这也说明它是最有发展的。

以上是总结结果，也许会有疑惑到底哪个加速器好呢？答：首先，用一定比不用好，其次每个加速器还有一些可以调优的参数，所以要根据系统环境而定，然后xcache和zendopcache这两款加速器的潜力还是很大的，可以多关注下。

## 1.7 补充一些知识

### 1.7.1 phpize是什么？

安装PHP扩展插件的时候，常常有这样一条命令：/application/php/bin/phpize，可能有人会问phpize是干嘛的？

事实上，phpize是用来扩展PHP扩展模块的，通过phpize可以建立PHP的外挂模块。比如你想在原来编译好的PHP中加入memcached等扩展模块，可以使用phpize工具。

下面是PHP官方的说明：<http://php.net/nanual/en/install.pecl.phpize.php>

那么，要如何使用phpize呢？

编译PHP后，其bin目录下会有phpize这个脚本文件，在编译要添加的扩展模块之前，执行以下phpize就可以了。比如现在想在PHP中加入memcached扩展模块：我们要做的只是执行memcached客户端软件的安装命令：

[root@lnmp tools]# cd /home/oldboy/tools/

[root@lnmp tools]# wget -q http://pecl.php.net/get/memcache-2.2.7.tgz

[root@lnmp tools]# tar zxf memcache-2.2.7.tgz

[root@lnmp tools]# cd memcache-2.2.7

[root@lnmp memcache-2.2.7]# /application/php/bin/phpize ##在执行configure前执行这个命令

[root@lnmp memcache-2.2.7]# ./configure --enable-memcache --with-php-config=/application/php/bin/php-config

[root@lnmp memcache-2.2.7]# make && make install

[root@lnmp memcache-2.2.7]# cd ../

[root@lnmp tools]# ll /application/php5.3.27/lib/php/extensions/no-debug-non-zts-20090626/

total 1780

-rwxr-xr-x 1 root root 417069 Jan 19 10:49 eaccelerator.so

-rwxr-xr-x 1 root root 246752 Jan 19 12:07 memcache.so ##最后生成了memcache.so模块就表示memcache扩展插件成功安装。

-rwxr-xr-x 1 root root 491814 Jan 19 11:48 opcache.so

-rwxr-xr-x 1 root root 658468 Jan 19 11:31 xcache.so

注意：./configure后面可以指定的是php-config文件的路径。

这样编译就完成了，还需要做的就是在php.ini文件中加入如下两行：

extension\_dir = “/application/php5.3.27/lib/php/extensions/no-debug-non-zts-20090626/”

extension = memcache.so

提示：extension\_dir的路径就是memcache.so模块文件所在的路径。

### 1.7.2 php指定MySQL的编译参数带来的问题

如果我们再编译php的时候指定了—with-mysql=mysqlnd，以及指定了—with-pdo-mysql=mysqlnd的参数进行编译，例如：

./configure\

--prefix=/application/php5.3.27\

--with-mysql=mysqlnd\

--with-mysqli=mysqlnd\

--with-pdo-mysql=mysqlnd\

省略其他参数..

则在生产中可能会遇到socket连接问题，解决方法是：在php.ini中添加

pdo\_mysql.default\_socket=/application/php5.3.27/tmp/mysql.sock